Contiguous group

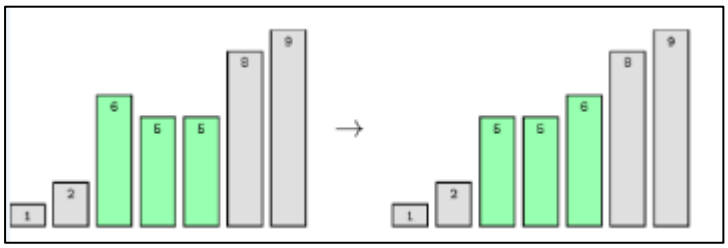
A class of N students is standing in a row. You want to take a picture of the students, but for aesthetic reasons you wish them to be arranged in order of height. To achieve this, you can tell a contiguous group of students to rearrange their positions. Your goal is to find the minimum number of students standing in a contiguous group who, after rearranging their positions, will cause the entire row to be ordered by height.

The current arrangement of the students is specified by a zero-indexed array A, of length N, in which element A[K] records the height of the student at position K. After the rearrangement, the students should be sorted non-decreasing order of height i.e. A[P] <= A[P+1] for every 0 <= P < N-1.

For example, if array A is as follows:

A[0] = 1
A[1] = 2
A[2] = 6
A[3] = 5
A[4] = 5
A[5] = 8
A[6] = 9

then the smallest group of students to be rearranged is A[2..4] of length 3. After rearranging this group, we obtain [1, 2, 5, 5, 6, 8, 9], which is in the correct order:



Any other rearrangement, involving contiguous group of fewer than three students, would not cause the resulting row to be sorted correctly.

Write a function:

```
object Solution { def solution(a: Array[Int]): Int }
```

that, given a zero-indexed array A describing a row of N students, returns the minimum number of students standing in a contiguous group that, when rearranged, causes the whole row to be ordered by height. If the row of students is already ordered correctly, the function should return 0.

For example, given the array A described above, the function should return 3.

Assume that:

- N is an integer within the range [1..100,000]
- each element of array A is an integer within the range [1..100,000,000]

Complexity:

- expected worst-case time complexity is O(N*log(N))

- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input array can be modified.